Amendments to the Specification:

Please replace paragraph **[0016]** with the following amended paragraph:

Figure 1B shows a more detailed block diagram of the system of ~~figure 1A~~ Figure 1A.

Please replace paragraph **[0024]** with the following amended paragraph:

Figure 8 shows a mapping of the function in ~~figure 7~~ Figure 7 using four-input look-up table circuits.

Please replace paragraph **[0026]** with the following amended paragraph:

Figure 10 shows a mapping of the function in ~~figure 9~~ Figure 9 using four-input look-up table circuits.

Please replace paragraph **[0028]** with the following amended paragraph:

Figure 12 shows a mapping of the netlist of the function in ~~figure 11~~ Figure 11 into four-input look-up tables.

Please replace paragraph **[0029]** with the following amended paragraph:

Figure 13 shows a structurally different netlist of the same function as that in ~~figure 11~~ Figure 11.

Please replace paragraph **[0030]** with the following amended paragraph:

Figure 14 shows a mapping of the netlist of the function in ~~figure 13~~ Figure 13 into four-input look-up tables.

Please replace paragraph **[0031]** with the following amended paragraph:

Figure 1A shows a system of the present invention for performing physical resynthesis. In an embodiment, software of the invention executes on a computer workstation system, such as shown in ~~figure 1A~~ Figure 1A. Figure 1A shows a computer system 1 that includes a monitor 3, screen 5, cabinet 7, keyboard 9, and mouse 11. Mouse 11 may have one or more buttons such as mouse buttons 13. Cabinet 07 houses familiar computer components, some of which are not shown, such as a processor, memory, mass storage devices 17, and the like. Mass storage devices 17 may include mass disk drives, floppy disks, Iomega ZIP™ disks, magnetic disks, fixed disks,

hard disks, CD-ROMs, recordable CDs, DVDs, DVD-R, DVD-RW, Flash and other nonvolatile solid-state storage, tape storage, reader, and other similar media, and combinations of these. A binary, machine-executable version, of the software of the present invention may be stored or reside on mass storage devices 17. Furthermore, the source code of the software of the present invention may also be stored or reside on mass storage devices 17 (*e.g.*, magnetic disk, tape, or CD-ROM).

Please replace paragraph **[0032]** with the following amended paragraph:

Furthermore, ~~figure 1B~~ Figure 1B shows a system block diagram of computer system 1 used to execute the software of the present invention. As in ~~figure 1A~~ Figure 1A, computer system 1 includes monitor 3, keyboard 9, and mass storage devices 17. Computer system 1 further includes subsystems such as central processor 22, system memory 24, input/output (I/O) controller 26, display adapter 28, serial or universal serial bus (USB) port 32, network interface 38, and speaker 40. The invention may also be used with computer systems with additional or fewer subsystems. For example, a computer system could include more than one processor 22 (*i.e.*, a multiprocessor system) or a system may include a cache memory.

Please replace paragraph **[0036]** with the following amended paragraph:

Figure 1C shows a block diagram of a digital system, which ~~the system~~ the invention may incorporate or operate on. The system may be provided on a single board, on multiple boards, or within multiple enclosures. Though embodiments of the present invention are useful in electronic and integrated circuits in general, they are particularly useful in programmable logic devices. Figure 1C illustrates a system 101 in which such a programmable logic device 121 may be utilized. Programmable logic devices or programmable logic integrated circuits are sometimes referred to as a PALs, PLAs, FPLAs, PLDs, CPLDs, EPLDs, EEPLDs, LCAs, or FPGAs and are well-known integrated circuits that provide the advantages of fixed integrated circuits with the flexibility of custom integrated circuits. Such devices allow a user to electrically program standard, off-the-shelf logic elements to meet a user's specific needs and are sold, *e.g.*, by Altera Corporation of San Jose, California. Programmable logic integrated circuits and their operation are well known to those of skill in the art.

Please replace paragraph **[0037]** with the following amended paragraph:

In the particular embodiment of ~~figure 1C~~ Figure 1C, a processing unit 101 is coupled to a memory 105 and an I/O 111, and incorporates a programmable logic device 121. PLD 121 may be specially coupled to memory 105 through connection 131 and to I/O 111 through connection 135. The system may be a programmed digital computer system, digital signal processing system, specialized digital switching network, or other processing system. Moreover, such systems may be designed for a wide variety of applications such as, merely by way of example, telecommunications systems, automotive systems, control systems, consumer electronics, personal computers, Internet communications and networking, and others.

Please replace paragraph **[0038]** with the following amended paragraph:

Processing unit 101 may direct data to an appropriate system component for processing or storage, execute a program stored in memory 105 or input using I/O 111, or other similar function. Processing unit 101 may be a central processing unit (CPU), microprocessor, floating point coprocessor, graphics coprocessor, hardware controller, microcontroller, programmable logic device programmed for use as a controller, network controller, or other processing unit. Furthermore, in many embodiments, there is often no need for a CPU. For example, instead of a CPU, one or more PLDs 121 may control the logical operations of the system. In an embodiment, PLD 121 acts as a reconfigurable processor, which can be reprogrammed as needed to handle a particular computing task. Alternately, programmable logic device 121 may include a processor. In some embodiments, processing unit 101 may even be a computer system. Memory 105 may be a random access memory (RAM), read only memory (ROM), fixed or flexible disk media, PC Card flash disk memory, tape, or any other storage retrieval means, or any combination of these storage retrieval means. PLD 121 may serve many different purposes within the system in ~~figure 1~~ Figure 1. PLD 121 may be a logical building block of processing unit 101, supporting its internal and external operations. PLD 121 is programmed to implement the logical functions necessary to carry on its particular role in system operation.

Please replace paragraph **[0041]** with the following amended paragraph:

LAB 200 has inputs and outputs (not shown) which may or may not be programmably connected to a global interconnect structure, comprising an array of horizontal interconnects 210 and vertical interconnects 220. Although shown as single lines in ~~figure 2~~ Figure 2, each set of interconnect lines may represent a plurality of signal conductors. The inputs and outputs of LAB 200 are programmably connectable to these sets of interconnect lines, such that multiple LABs 200 may be connected and combined to implement larger, more complex logic functions than can be realized using a single LAB 200.

Please replace paragraph **[0043]** with the following amended paragraph:

The programmable logic architecture in ~~figure 2~~ Figure 2 further shows at the peripheries of the chip, input and output circuits 230. Input and output circuits 230 are for interfacing the PLD to external, off-chip circuitry. Some or all of these input and output circuits 230 may be consistent with embodiments of the present invention. Figure 2 shows thirty-two input and output circuits 230; however, a programmable logic integrated circuit may contain any number of input and output circuits, more or less than the number depicted. As discussed above, some of these input-output drivers may be shared between the embedded processor and programmable logic portions. Each input and output circuit 230 is configurable for use as an input driver, output driver, or bidirectional driver. In other embodiments of a programmable logic integrated circuit, the input and output circuits may be embedded with the integrated circuit core itself. This embedded placement of the input and output circuits may be used with flip chip packaging and will minimize the parasitics of routing the signals to input and output circuits.

Please replace paragraph **[0044]** with the following amended paragraph:

Figure 3 shows a simplified block diagram of LAB 200 of ~~figure 2~~ Figure 2. LAB 200 is comprised of a varying number of logic elements (LEs) 300, sometimes referred to as "logic cells" or LCELLs, and a local (or internal) interconnect structure 310. LAB 200 has eight LEs 300, but LAB 200 may have any number of LEs, more or less than eight.

Please replace paragraph **[0047]** with the following amended paragraph:

Figure 4 shows an example for a typical programmable logic architecture. The architecture in ~~figure 4~~ Figure 4 further includes embedded array blocks (EABs). EABs contain user memory, a flexible block of RAM. The embedded array blocks can be configured as FIFOs acting as frequency translators and serial to parallel converters for interfacing between high-speed input and outputs and the core circuits including the logic array blocks.

Please replace paragraph **[0048]** with the following amended paragraph:

Figure 5 shows an example of a flow diagram of a electronic design automation (EDA) or computer-aided design (CAD) tool used in the design of integrated circuits including microprocessors, ASICS, memories, FPGAs, PLDs, and others. In a specific implementation, this flow is used to configure a programmable logic integrated circuit. As discussed above, a user typically programs a programmable logic integrated with the user's desired logic. Figure 5 is an example of one technique of designing and implementing logic for a programmable logic integrated circuit. At Altera, a particular implementation of the technique of this flow is implemented using a software system referred to as Quartus. This is just an example. One could draw the flow with more or fewer steps and targeting a specific or more general device hardware. The most common implementation of this flow would be as a computer program executing as part of a system similar to that shown in ~~figure 1~~ Figure 1, though other implementations are possible.

Please replace paragraph **[0066]** with the following amended paragraph:

Each of these steps in the ~~figure 5~~ Figure 5 flow above is commonly broken down into further steps. In an embodiment, the invention is a resynthesis technique and may add additional steps to the flow or may be additional steps within one of the existing steps.

Please replace paragraph **[0069]** with the following amended paragraph:

In a specific embodiment, this invention is a method of incorporating late-stage synthesis operations (i.e., technology mapping algorithms) into early-stage synthesis procedures (i.e., general logic minimization and synthesis algorithms) for estimating delay and area of final

compiled designs. Referring to ~~figure 5~~ Figure 5, the technology mapping algorithms typically are utilized at the end of a synthesis step 504 or in the place and route step 506.

Please replace paragraph **[0074]** with the following amended paragraph:

This patent describes a way to better estimate the number of logic cells and the circuit depth during early stages of synthesis. Synthesis translates ~~the a~~ a users' HDL source into a netlist of logic cells (lcells), I/Os, and hard blocks like DSP blocks and memory blocks. Traditionally, as shown in ~~figure 6~~ Figure 6, synthesis is divided into four independent stages: (1) extraction 602, (2) high-level optimization 606, (3) logic minimization 610, and (4) technology mapping 616. As discussed above, these stages may be allocated all within synthesis 504 of ~~figure 5~~ Figure 5 (as discussed in this application), or may be divided among different steps of the ~~figure 5~~ Figure 5 flow, such as synthesis and place and route 506. These four stages may be divided or allocated in any way within the ~~figure 5~~ Figure 5 CAD flow.

Please replace paragraph **[0079]** with the following amended paragraph:

For instance, consider the 4-to-1 multiplexer function: out = s0'sl'd0 + s0sl'dl + s0'sld2 + s0sld3. Figure 7 shows a simple gate representation of this function with nine two-input gates. This is the optimal number of two input gates that one can obtain for any netlist for this function. If this netlist is mapped to four-input LUTs, one obtains the netlist in ~~figure 8~~ Figure 8. Note that this netlist has three LUTs.

Please replace paragraph **[0080]** with the following amended paragraph:

Figure 9 shows another representation of the four-to-one multiplexer that uses ten two-input gates. However, when we map this netlist to four-input LUTs, we obtain a netlist with two four-input LUTs as shown in Figure 10. So as can be seen, if a decomposition with a minimum number of gates for this function (as in ~~figure 7~~ Figure 7) is selected, one would not find the mapping with minimum number of LUTs (in ~~figure 10~~ Figure 10).

Please replace paragraph **[0081]** with the following amended paragraph:

Figures 11 and 13 represent the same function but with structurally slightly different netlists. Both netlists have exactly the same number of two-input gates and the same gate depth.

The netlist of ~~figure 11~~ Figure 11 may be considered better for circuit depth, since it has fewer two-input gates of maximum gate depth.

Please replace paragraph **[0082]** with the following amended paragraph:

Figures 12 and 14 show mappings to four-input LUTs of the netlists in ~~figures 11~~ Figures 11 and 13, respectively. As can be seen the mapping of Figure 14 is far superior over the mapping of ~~figure 12~~ Figure 12: it has smaller circuit depth and uses fewer LUTs.

Please replace paragraph **[0085]** with the following amended paragraph:

The main input to EstMap is a netlist of synthesis gates (i.e., gates results from synthesis operation, such as those gates in ~~figure 7~~ Figure 7 or 9) ~~(INVENTORS.IS THIS CORRECT?)~~,which may either be a complete netlist with inputs and outputs, or a subnetlist of a complete netlist, defined by boundary gates. EstMap also has a Boolean parameter which tells it to map for best area or for best speed (similar to a logic option that the customer can set). Output of EstMap is the mapped version of the netlist and information about the number LUTs used in the mapping and the circuit depth of the mapped netlist. EstMap is a very fast algorithm, which means it can be used many times during synthesis without significantly increasing the total compile time.

Please replace paragraph **[0086]** with the following amended paragraph:

Referring to ~~figure 6~~ Figure 6, EstMap is a reduced version of the technology mapping routine found in technology mapping step 616. This EstMap routine is used earlier in the synthesis flow in high-level optimization step 606 (instead of only during technology mapping step 616) to determine which alternative of the synthesis gate extraction provides the best results for delay or area, or both. This particular alternative of the synthesis gate extraction is then used for further optimization operations and logic minimization operations 610.